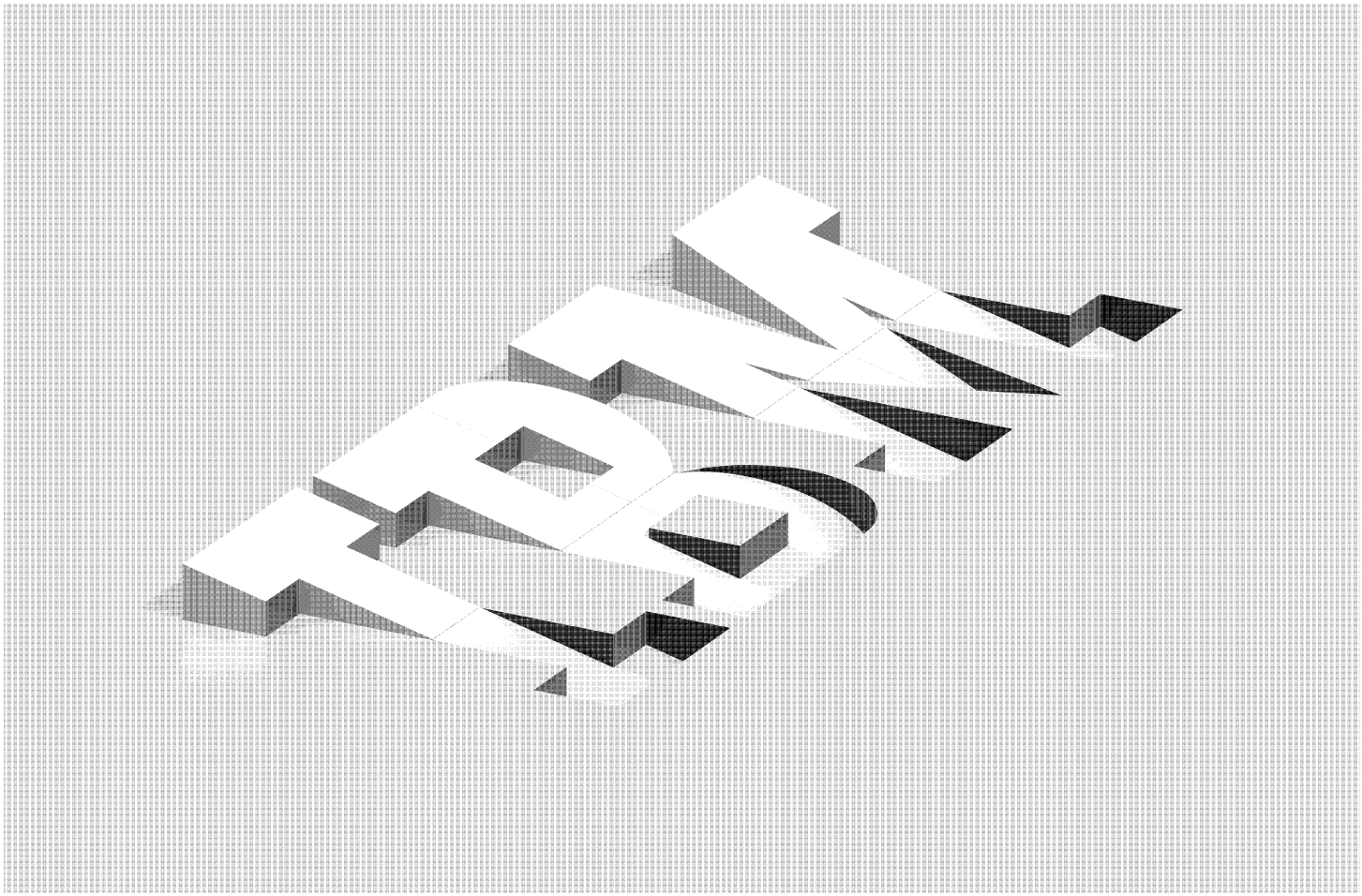
EXHIBIT 54

IBM Cloud

Kubernetes service

Product guide



Edition notices

This PDF was created on 2025-05-23 as a supplement to *Kubernetes service* in the IBM Cloud docs. It might not be a complete set of information or the latest version. For the latest information, see the IBM Cloud documentation at https://cloud.ibm.com/docs/containers.

Getting started with IBM Cloud Kubernetes Service

IBM Cloud Kubernetes Service is a managed Kubernetes service to create your own cluster of compute hosts where you can deploy and manage containerized apps on IBM Cloud. Combined with an intuitive user experience, built-in security and isolation, and advanced tools to secure, manage, and monitor your cluster workloads, you can rapidly deliver highly available and secure containerized apps in the public cloud.

Complete the following steps to get familiar with the basics, understand the service components, create your first cluster, and deploy a starter app.

Step 1: Review the basics

Get an overview of the service by reviewing the concepts, terms, and benefits. For more information, see Understanding IBM Cloud Kubernetes Service.

Already familiar with containers and IBM Cloud Kubernetes Service? Continue to the next step to prepare your account for creating clusters.

Step 2: Prepare your account

To set up your IBM Cloud account so that you can create clusters, see Preparing your account to create clusters.

If you've already prepared your account and you're ready to create a cluster, continue to the next step.

Step 3: Create a cluster environment strategy

Review the decision points in the Creating a cluster environment strategy doc to begin designing your setup.

Tip: Not sure where to start? Try following a tutorial in the next step.

Step 4: Create a cluster

Follow a tutorial, or set up your own custom cluster environment. Review the following table for your deployment options.

Type	Level	Time	Description
Tutorial	Beginner	1 hour	Follow the steps in this tutorial to create your own Virtual Private Cloud (VPC), then create an IBM Cloud Kubernetes Service cluster by using the CLI. For more information, see Create a cluster in your own Virtual Private Cloud.
Custom deployment	Intermediate	1-3 hours	Create a custom cluster on Classic infrastructure.
Custom deployment	Intermediate	1-3 hours	Create a custom cluster on VPC infrastructure.

Options for creating a cluster

Already have a cluster? Continue to the next step to deploy a sample app!

Step 5: Deploy a sample app

After you create a cluster, deploy your first app. You can use a sample `websphere-liberty` Java application server that IBM provides and deploy the app to your cluster by using the Kubernetes dashboard.

1. Select your cluster from the cluster list.
2. Click **Kubernetes dashboard**.
3. Click the **Create new resource** icon (`+`) and select the **Create from form** tab.
    a. Enter a name for your app, such as `liberty`.
    b. Enter `websphere-liberty` for your container image. Remember that your cluster's VPC subnet must have a public gateway so that the cluster can pull an image from DockerHub.
    c. Enter the number of pods for your app deployment, such as `1`.
    d. From the **Service** drop-down menu, select **External** to create a `LoadBalancer` service that external users can use to access the app. Configure the external service as follows.

- **Port**: `80`
- **Target port**: `9080`
- **Protocol**: `TCP`

4. Click **Deploy**. During the deployment, the cluster downloads the `websphere-liberty` container image from Docker Hub and deploys the app in your cluster. Your app is exposed by a Layer 4, version 1.0 network load balancer (NLB) so that it can be accessed by other users internally or externally. For other ways to expose an app such as Ingress, see Planning in-cluster and external networking for apps.
5. From the **Pods** menu, click your `liberty` pod and check that its status is **Running**.
6. From the **Services** menu, click the **External Endpoint** of your `liberty` service. For example, `169.xx.xxx.xxx:80` for classic clusters or `http://<hash>-<region>.lb.appdomain.cloud/` for VPC clusters. The **Welcome to Liberty** page is displayed.

Great job! You just deployed your first app in your Kubernetes cluster.

What's next?

Check out the curated learning paths

- Learning path for administrators.
- Learning path for developers.

Understanding IBM Cloud Kubernetes Service

Learn more about IBM Cloud® Kubernetes Service, its capabilities, and the options that are available to you to customize the cluster to your needs.

IBM Cloud Kubernetes Service is a managed offering to create your own Kubernetes cluster of compute hosts to deploy and manage containerized apps on IBM Cloud. As a certified Kubernetes provider, IBM Cloud Kubernetes Service provides intelligent scheduling, self-healing, horizontal scaling, service discovery and load balancing, automated rollouts and rollbacks, and secret and configuration management for your apps. Combined with an intuitive user experience, built-in security and isolation, and advanced tools to secure, manage, and monitor your cluster workloads, you can rapidly deliver highly available and secure containerized apps in the public cloud.

Review frequently asked questions and key technologies that IBM Cloud Kubernetes Service uses.

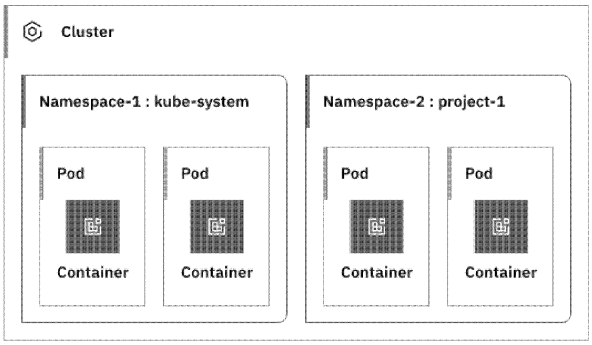
What is Kubernetes?

Kubernetes is an open source platform for managing containerized workloads and services across multiple hosts, and offers management tools for deploying, automating, monitoring, and scaling containerized apps with minimal to no manual intervention.

This badge indicates Kubernetes certification for IBM Cloud Container Service.

The open source project, Kubernetes, combines running a containerized infrastructure with production workloads, open source contributions, and Docker container management tools. The Kubernetes infrastructure provides an isolated and secure app platform for managing containers that is portable, extensible, and self-healing in case of a failover. For more information, see What is Kubernetes?.

Learn more about the key concepts of Kubernetes as illustrated in the following image.



A description of key concepts for Kubernetes

Account

Your account refers to your IBM Cloud account.

Cluster, worker pool, and worker node

A Kubernetes cluster consists of a master and one or more compute hosts that are called worker nodes. Worker nodes are organized into worker pools of the same flavor, or profile of CPU, memory, operating system, attached disks, and other properties. The worker nodes correspond to the Kubernetes `Node` resource, and are managed by a Kubernetes master that centrally controls and monitors all Kubernetes resources in the cluster. So when you deploy the resources for a containerized app, the Kubernetes master decides which worker node to deploy those resources on, accounting for the deployment requirements and available capacity in the cluster. Kubernetes resources include services, deployments, and pods.

Namespace

Kubernetes namespaces are a way to divide your cluster resources into separate areas that you can deploy apps and restrict access to, such as if you want to share the cluster with multiple teams. For example, system resources that are configured for you are kept in separate namespaces like `kube-system` or `ibm-system`. If you don't designate a namespace when you create a Kubernetes resource, the resource is automatically created in the `default` namespace.

Service

A service is a Kubernetes resource that groups a set of pods and provides network connectivity to these pods without exposing the actual private IP address of each pod. You can use a service to make your app available within your cluster or to the public internet.

Deployment

A deployment is a Kubernetes resource where you might specify information about other resources or capabilities that are required to run your app, such as services, persistent storage, or annotations. You document a deployment in a configuration YAML file, and then apply it to the cluster. The Kubernetes master configures the resources and deploys containers into pods on the worker nodes with available capacity.

Define update strategies for your app, including the number of pods that you want to add during a rolling update and the number of pods that can be unavailable at a time. When you perform a rolling update, the deployment checks whether the update is working and stops the rollout when failures are detected.

A deployment is just one type of workload controller that you can use to manage pods. For help choosing among your options, see What type of Kubernetes objects can I make for my app?. For more information about deployments, see the Kubernetes documentation.

Pod

Every containerized app that is deployed into a cluster is deployed, run, and managed by a Kubernetes resource called a pod. Pods represent small deployable units in a Kubernetes cluster and are used to group the containers that must be treated as a single unit. Usually, each container is deployed in its own pod. However, an app might require a container and other helper containers to be deployed into one pod so that those containers can be addressed by using the same private IP address.

App

An app might refer to a complete app or a component of an app. You might deploy components of an app in separate pods or separate worker nodes. For more information, see Planning app deployments and Developing Kubernetes-native apps.

Tip: To dive deeper into Kubernetes, see the Kubernetes documentation.

What are containers?

Containers provide a standard way to package your application's code, configurations, and dependencies into a single unit that can run as a resource-isolated process on a compute server. To run your app on IBM Cloud, you must first containerize your app by creating a container image that you store in a container registry.

Review the following terms to get more familiar with the concepts.

Container

A container is an app that is packaged with all its dependencies so that the app can be moved between environments and run without changes. Unlike virtual machines, containers don't virtualize a device, its operating system, and the underlying hardware. Only the app code, run time, system tools, libraries, and settings are packaged inside the container. Containers run as isolated processes on compute hosts and share the host operating system and its hardware resources. This approach makes a container more lightweight, portable, and efficient than a virtual machine.

Image

A container image is a package that includes the files, configuration settings, and libraries to run a container. An image is built from a text file called a Dockerfile. Dockerfiles define how to build the image and which artifacts to include in it. The artifacts that are included in a container consist of the app code, configuration settings, and any dependencies.

Registry

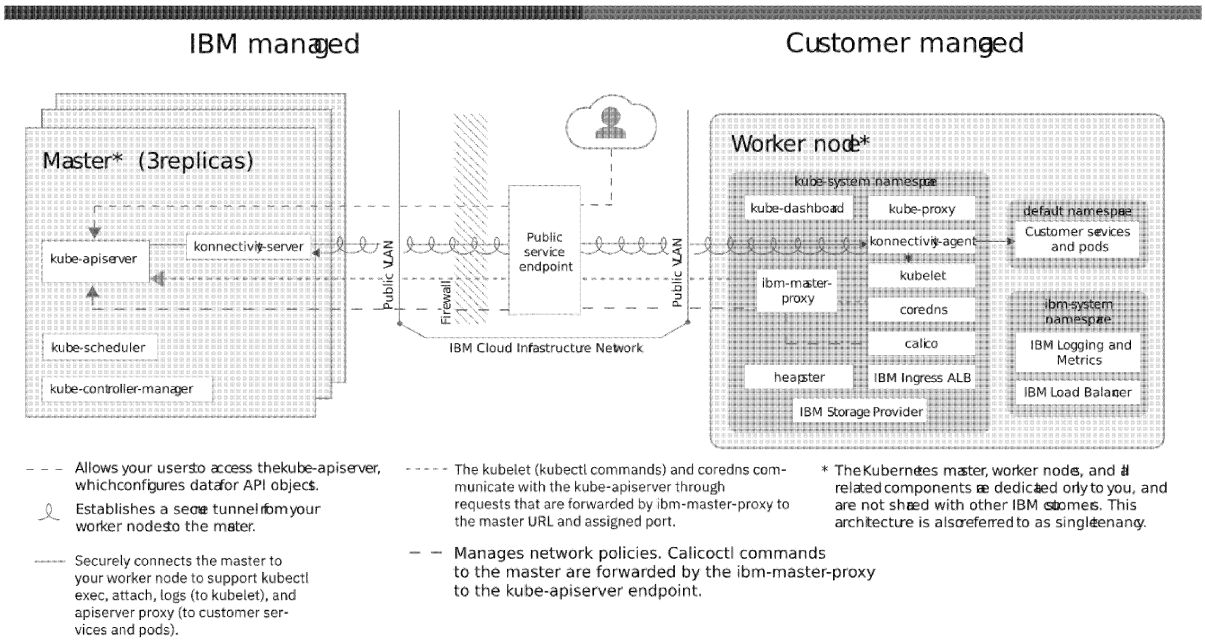
An image registry is a place to store, retrieve, and share container images. Registries can either be publicly available to anyone or privately available to a small group of users. When it comes to enterprise applications, use a private registry like IBM Cloud to protect your images from being used by unauthorized users.

What compute host infrastructure does IBM Cloud Kubernetes Service offer?

With IBM Cloud® Kubernetes Service, you can create a cluster by using infrastructure from the following providers. All the worker nodes in a cluster must be from the same provider.

Component	Description
Overview	Create clusters on virtual servers in your own Virtual Private Cloud (VPC).
Supported container platforms	Red Hat OpenShift or Kubernetes
Compute and worker node resources	Worker nodes are created as virtual machines by using either shared infrastructure or dedicated hosts. Unlike classic clusters, VPC cluster worker nodes on shared hardware don't appear in your infrastructure portal or a separate infrastructure bill. Instead, you manage all maintenance and billing activity for the worker nodes through IBM Cloud Kubernetes Service. Your worker node instances are connected to certain VPC instances that do reside in your infrastructure account, such as the VPC subnet or storage volumes. For dedicated hosts, the dedicated host price covers the vCPU, memory, and any instance storage to be used by any workers placed on the host. Note that all Intel® x86-64 servers have Hyper-Threading enabled by default. For more information, see Intel Hyper-Threading Technology.
Security	Clusters on shared hardware run in an isolated environment in the public cloud. Clusters on dedicated hosts do not run in a shared environment, instead only your clusters are present on your hosts. Network access control lists protect the subnets that provide the floating IPs for your worker nodes.
High availability	The master includes three replicas for high availability. Further, if you create your cluster in a multizone metro, the master replicas are spread across zones and you can also spread your worker pools across zones.
Reservations	Reservations aren't available for VPC.
Cluster administration	VPC clusters can't be reloaded or updated. Instead, use the `worker replace --update` CLI or API operation to replace worker nodes that are outdated or in a troubled state.
Cluster networking	Unlike classic infrastructure, the worker nodes of your VPC cluster are attached to VPC subnets and assigned private IP addresses. The worker nodes are not connected to the public network, which instead is accessed through a public gateway, floating IP, or VPN gateway. For more information, see Overview of VPC networking in IBM Cloud Kubernetes Service.
Apps and container platform	You can choose to create community Kubernetes or Red Hat OpenShift clusters to manage your containerized apps. Your app build processes don't differ because of the infrastructure provider, but how you expose the app does.
App networking	All pods that are deployed to a worker node are assigned a private IP address in the 172.30.0.0/16 range and are routed between worker nodes on the worker node private IP address of the private VPC subnet. To expose the app on the public network, you can create a Kubernetes `LoadBalancer` service, which provisions a VPC load balancer and public hostname address for your worker nodes. For more information, see Exposing apps with VPC load balancers.
Storage	You can choose from non-persistent and persistent storage solutions such as file, block, object, and software-defined storage. For more information, see Planning highly available persistent storage.
User access	You can use IBM Cloud IAM access policies to authorize users to create infrastructure, manage your cluster, and access cluster resources. The cluster can be in a different resource group than the VPC.
Integrations	VPC supports a select list of supported IBM Cloud services, add-ons, and third-party integrations. For a list, see Supported IBM Cloud and third-party integrations.
Locations and versions	VPC clusters are available worldwide in the multizone location.
Service interface	VPC clusters are supported by the next version (v2) of the IBM Cloud Kubernetes Service API, and you can manage your VPC clusters through the same CLI and console as classic clusters.

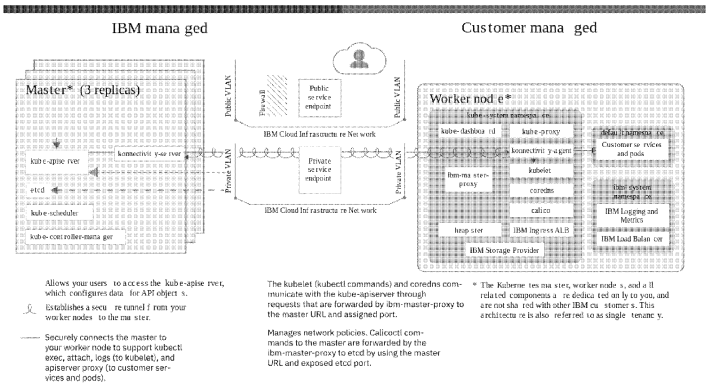
endpoint is enabled.



Cluster architecture when only the public cloud service endpoint is enabled

VRF-enabled account with private and public cloud service endpoints

The following image shows the components of your cluster and how they interact in a VRF-enabled account when the public and private cloud service endpoints are enabled.



Cluster architecture when public and private cloud service endpoints are enabled

Kubernetes master components

The Kubernetes master is tasked with managing all compute, network, and storage resources in the cluster, and ensures that your containerized apps and services are equally deployed to the worker nodes in the cluster. Depending on how you configure your app and services the master determines the worker node that has sufficient resources to fulfill the app's requirements.

Note: The Kubernetes master and all the master components are dedicated only to you, and are not shared with other IBM customers.

The following table describes the components of the Kubernetes master.

`kube-apiserver`

The Kubernetes API server serves as the main entry point for all cluster management requests from the worker node to the Kubernetes master. The Kubernetes API server validates and processes requests that change the state of Kubernetes resources, such as pods or services, and stores this state in etcd.

`konnectivity-server`

The Konnectivity server works with the Konnectivity agent to securely connect the master to the worker node. This connection supports `apiserver proxy` calls to your pods and services, and `kubectl exec`, `attach`, and `logs` calls to the kubelet.

`etcd`

`etcd` is a highly available key value store that stores the state of all Kubernetes resources of a cluster, such as services, deployments, and pods. Data in etcd is backed up to an encrypted storage instance that IBM manages.

`kube-scheduler`

The Kubernetes scheduler watches for newly created pods and decides where to deploy them based on capacity, performance needs, policy constraints, anti-affinity specifications, and workload requirements. If no worker node can be found that matches the requirements, the pod is not deployed in the cluster.

`kube-controller-manager`

The Kubernetes controller manager is a daemon that watches the state of cluster resources, such as replica sets. When the state of a resource changes, for example if a pod in a replica set goes down, the controller manager initiates correcting actions to achieve the required state.

Worker node components

Each worker node is a physical machine (bare metal) or a virtual machine that runs on physical hardware in the cloud environment. When you provision a worker node, you determine the resources that are available to the containers that are hosted on that worker node. Out of the box, your worker nodes are set up with an IBM-managed container runtime, separate compute resources, networking, and a volume service. The built-in security features provide isolation, resource management capabilities, and worker node security compliance.

Note: The worker nodes and all the worker node components are dedicated only to you, and are not shared with other IBM customers. However, if you use a worker node virtual machine, the underlying hardware might be shared with other customers depending on the level of hardware isolation that you choose.

Note: Modifying default worker node components such as the `kubelet` is not supported and might cause unexpected results.

The following tables describe the components of a worker node.

`kube-system` namespace

`ibm-master-proxy`

The `ibm-master-proxy` forwards requests from the worker node to the IP addresses of the highly available master replicas. In single zone clusters, the master has three replicas on separate hosts with one master IP address and domain name. For clusters that are in a multizone-capable zone, the master has three replicas that are spread across zones. As such, each master has its own IP address that is registered with DNS, with one domain name for the entire cluster master.

`konnectivity-agent`

The Konnectivity agent works with the Konnectivity server to securely connect the master to the worker node. This connection supports `apiserver proxy` calls to your pods and services, and `kubectl exec`, `attach`, and `logs` calls to the kubelet.

`kubelet`

The kubelet is a pod that runs on every worker node and is responsible for monitoring the health of pods that run on the worker node and for watching the events that the Kubernetes API server sends. Based on the events, the kubelet creates or removes pods, ensures liveness and readiness probes, and reports back the status of the pods to the Kubernetes API server.

`coredns`

By default, Kubernetes schedules a CoreDNS pod (or KubeDNS pod in version 1.12 and earlier) and service on the cluster. Containers automatically use the DNS service's IP to resolve DNS names in their searches for other pods and services.

`calico`

Calico manages network policies for your cluster, and comprises a few components as follows.

`calico-cni`: The Calico container network interface (CNI) manages the network connectivity of containers and removes allocated resources when a container is deleted.

`calico-ipam`: The Calico IPAM manages IP address assignment for containers.

`calico-node`: The Calico node is a container that bundles together the various components that are required for networking containers with Calico.

`calico-policy-controller` : The Calico policy controller watches inbound and outbound network traffic for compliance with set network policies. If the traffic is not allowed in the cluster, access to the cluster is blocked. The Calico policy controller is also used to create and set network policies for a cluster.

`kube-proxy`

The Kubernetes network proxy is a daemon that runs on every worker node and that forwards or load balances TCP and UDP network traffic for services that run in the cluster.

`kube-dashboard`

The Kubernetes dashboard is a web-based GUI that allows users to manage and troubleshoot the cluster and applications that run in the cluster.

`heapster`

Heapster is a cluster-wide aggregator of monitoring and event data. The Heapster pod discovers all nodes in the cluster and queries usage information from each node's kubelet. You can find utilization graphs in the Kubernetes dashboard.

Ingress ALB

Ingress is a Kubernetes service that you can use to balance network traffic workloads in your cluster by forwarding public or private requests to multiple apps in your cluster. To expose your apps over the public or private network, you must create an Ingress resource to register your apps with the Ingress application load balancer (ALB). Multiple apps can then be accessed by using a single URL or IP address.

Storage provider

Every cluster is set up with a plug-in to provision file storage. You can choose to install other add-ons, such as block storage.

`ibm-system` namespace

Logging and metrics

You can use the IBM Cloud Logs and IBM Cloud® Monitoring services to expand your collection and retention capabilities when working with logs and metrics. Load balancer

A load balancer is a Kubernetes service that can be used to balance network traffic workloads in your cluster by forwarding public or private requests to an app.

`default` namespace

App pods and services

In the `default` namespace or in namespaces that you create, you can deploy apps in pods and services to communicate with those pods.

VPC cluster

The following diagram and table describe the default components that are set up in an IBM Cloud Kubernetes Service VPC cluster architecture.

Note: The following architectural overviews are specific to the VPC infrastructure provider. For an architectural overview for the classic infrastructure provider, see Classic cluster architecture.

2. Copy the YAML file and in the `spec.volumes` section, change the name of the secret that you want to use.
3. Apply the change in your cluster.

   ```
   $ kubectl apply -f pod.yaml
   ```

   ```
   $ kubectl apply -f deployment.yaml
   ```

4. Verify that a new pod is created with the updated volume specification.

   ```
   $ kubectl get pods
   ```

   ```
   $ kubectl describe pod <pod_name>
   ```

5. Remove the secret.

   ```
   $ kubectl delete secret <secret_name> -n <namespace>
   ```

6. Verify that your secret is removed.

   ```
   $ kubectl get secrets -n <namespace>
   ```

7. Optional. Remove the IBM Cloud service instance.

   ```
   $ ibmcloud resource service-instance-delete <service_name>
   ```

Managing cluster costs

Understanding costs for your clusters

With IBM Cloud®, you can plan for, estimate, review, and modify your cluster environment to control costs. Just by using a managed service like IBM Cloud® Kubernetes Service, you are saving many expenses that are associated with managing, updating, and maintaining an infrastructure environment.

Understanding costs by component

With IBM Cloud Kubernetes Service clusters, you can use IBM Cloud infrastructure compute, networking, and storage resources with platform services such as Watson AI or Compose Database-as-a-Service. Each resource might entail its own charges that can be fixed, metered, tiered, or reserved and billed by various incremental rates such as monthly or hourly.

Note: Monthly resources are billed based on the first of the month for usage in the preceding month. If you order a monthly resource in the middle of the month, you are charged a prorated amount for that month. However, if you cancel a resource in the middle of the month, you are still charged the full amount for the monthly resource.

Worker nodes

Clusters can have two main types of worker nodes: virtual or physical (bare metal) machines. Flavor (machine type) availability and pricing varies by the zone that you deploy your cluster to.

When do worker nodes begin to incur charges?**

Worker nodes begin to incur charges after they complete the `provisioning` state and continue until you delete the worker nodes and they complete the `deleting` state. For more information, see Worker node states.

What is the difference between virtual and physical machines?

Virtual machines feature greater flexibility, quicker provisioning times, and more automatic scalability features than bare metal, at a more cost-effective price than bare-metal. However, VMs have a performance tradeoff when compared to bare metal specs, such as networking Gbps, RAM and memory thresholds, and storage options. Keep in mind these factors that impact your VM costs.

- **Shared versus dedicated**: If you share the underlying hardware of the VM, the cost is less than dedicated hardware, but the physical resources are not dedicated to your VM.
- **Hourly billing only**: Hourly billing offers more flexibility to order and cancel VMs quickly. You are charged an hourly rate that is metered for only the time that that the worker node is provisioned. The time is not rounded up or down to the nearest hour, but is metered in minutes and charged at the hourly rate. For example, if your worker node is provisioned for 90 minutes, you are charged the hourly rate for 1.5 hours, not 2 hours.
- **Tiered hours per month**: The pricing is billed hourly in graduated tiered. As your VM remains ordered for a tier of hours within a billing month, the hourly rate that you are charged lowers. The tiers of hours are as follows:
  - 0 - 150 hours
  - 151 - 290 hours
  - 291 - 540 hours
  - 541+ hours

Physical machines, or bare metal, (not available for VPC clusters) yield high-performance benefits for workloads such as data, GPU, and AI. Additionally, all the hardware resources are dedicated to your workloads, so you don't have "noisy neighbors". Keep in mind these factors that impact your bare metal costs.

- **Monthly billing only**: All bare metals are charged monthly.
- **Longer ordering process**: After you order or cancel a bare metal server, the process is completed manually in your IBM Cloud infrastructure account. Therefore, it can take more than one business day to complete.

Note: VPC Generation 2 only: Prices vary by region where the underlying worker node infrastructure resides, and you can get sustained usage discounts. For more information, see What are the regional uplift charges and sustained usage discounts for VPC worker nodes? .

Public bandwidth

Bandwidth refers to the public data transfer of inbound and outbound network traffic, both to and from IBM Cloud resources in data centers around the

globe.

Classic clusters: Public bandwidth is charged per GB. You can review your current bandwidth summary by logging into the IBM Cloud console, from the menu ☰ selecting **Infrastructure** > **Classic Infrastructure**, and then selecting the **Network > Bandwidth > Summary** page.

Review the following factors that impact public bandwidth charges:

- **Location**: As with worker nodes, charges vary depending on the zone that your resources are deployed in.
- **Pay-As-You-Go for VM**: Because VMs are billed at an hourly rate, your VM worker node machines have a Pay-As-You-Go allocation of outbound networking based on GB usage.
- **Included bandwidth and tiered packages for BM** : Bare metal worker nodes might come with a certain allocation of outbound networking per month that varies by geography: 20 TB for North America and Europe, or 5 TB for Asia Pacific and South America. After you exceed your included bandwidth, you are charged according to a tiered usage scheme for your geography. If you exceed a tier allotment, you might also be charged a standard data transfer fee.

VPC clusters: For more information about how internet data transfer works in your Virtual Private Cloud, see Pricing for VPC.

Subnet IP addresses

Subnets for IBM Cloud Kubernetes Service clusters vary by infrastructure provider.

Classic clusters: When you create a standard cluster, a portable public subnet with 8 public IP addresses is ordered and charged to your account monthly. For pricing information, see the flag, and then reuse your subnets.

VPC clusters: For more information about charges for floating IPs and other networking costs, see Pricing for VPC.

Multizone load balancer

When you create a multizone cluster or add zones to a single zone cluster, you must have a load balancer to health check Ingress and load balancer IP addresses in each zone, and forward requests to your apps across zones in the region.

The type of load balancer that is automatically created varies depending on the type of cluster.

- **Classic clusters**: An Akamai MZLB is automatically created for each multizone cluster. You can view the hourly rate in the pricing summary when you create the cluster.
- **VPC clusters**: A Load Balancer for VPC is automatically created in your VPC for your cluster. For cost information, see Pricing for Load Balancer for VPC.

Storage

When you provision storage, you can choose the storage type and storage class that is correct for your use case. Charges vary depending on the type of storage, the location, and the specs of the storage instance. Some storage solutions, such as file and block storage offer hourly and monthly rates that you can choose from.

To choose the correct storage solution, see Planning highly available persistent storage. For more information, see:

- File Storage for Classic
- Block Storage for Classic
- File Storage for VPC
- Block Storage for VPC
- IBM Cloud Object Storage
- Portworx Enterprise pricing

IBM Cloud services

Each service that you integrate with your cluster has its own pricing model. Review each product documentation and use the IBM Cloud console to estimate costs.

Operators and other third-party integrations

Operators and other third-party integrations are a convenient way to add services to your cluster from community, third-party, your own, or other providers. Keep in mind that you are responsible for additional charges and how these services operate in your cluster, from deployment and maintenance to integration with your apps. If you have issues with an operator or third-party integration, work with the appropriate provider to troubleshoot the issue.

VPC worker nodes